R E P O R T R E S U M E S

ED 019 632 AL 000 577

SYSTEM DESIGN FOR COMPUTATIONAL LINGUISTICS.

BY- JONAS, RONALD W.
TEXAS UNIV., AUSTIN, LINGUISTICS RES. CTR.

PUB DATE JUN 67

EDRS PRICE MF-$0.25 HC-$1.44 34P.

DESCRIPTORS- *COMPUTATIONAL LINGUISTICS, *LANGUAGE RESEARCH, PROGRAMING, INFORMATION STORAGE, INFORMATION RETRIEVAL, DEEP STRUCTURE, SURFACE STRUCTURE, *TRANSFORMATION GENERATIVE GRAMMAR, TRANSFORMATIONS (LANGUAGE), SYNTAX, *CONTEXT FREE GRAMMAR, *MACHINE TRANSLATION, SEMANTICS,

THIS PAPER IS THE FIRST IN A SERIES PRESENTING THE FEATURES OF SYSTEMS DESIGNED FOR COMPUTATIONAL LINGUISTICS AT THE LINGUISTICS RESEARCH CENTER (LRC) OF THE UNIVERSITY OF TEXAS AT AUSTIN. ONGOING RESEARCH IS EXPANDING THE APPLICATION OF THESE SYSTEMS TO INCLUDE NOT ONLY NATURAL LANGUAGE BUT PROGRAMING LANGUAGES AS WELL. THIS PAPER DISCUSSES CURRENT PARSERS, SYNTHESIZERS, AND TRANSLATORS, GIVING FULLEST TREATMENT TO (1) TYPES OF GRAMMAR MOST SUITABLE FOR LANGUAGE DESCRIPTION, (2) GRAMMATICAL DESCRIPTION OF DISCONTINUITIES, AND (3) INTERLINGUAL MAPPING. ALTERNATIVES ARE SUGGESTED AND CHOICES ARE MADE AS TO WHICH FEATURES ARE MOST LIKELY TO BENEFIT RESEARCH IN COMPUTATIONAL LINGUISTICS. FURTHER INFORMATION ON THIS PAPER AND CURRENT RESEARCH AT LRC MAY BE OBTAINED FROM THE LINGUISTICS RESEARCH CENTER, THE UNIVERSITY OF TEXAS AT AUSTIN, BOX 7247, UNIVERSITY STATION, AUSTIN, TEXAS 78712. (AUTHOR/DO)

ED019632

POSITION OR POLICY.

SYSTEM DESIGN FOR COMPUTATIONAL LINGUISTICS

Ronald W. Jonas

"PERMISSION TO REPRODUCE THIS [redacted] MATERIAL HAS BEEN GRANTED BY RONALD W. JONAS

TO ERIC AND ORGANIZATIONS OPERATING

THIS IS A WORKING PAPER □ IT MAY BE EXPANDED, MODIFIED OR WITHDRAWN AT ANY TIME □ THE VIEWS, CONCLUSIONS, AND RECOMMENDATIONS EXPRESSED HEREIN DO NOT NECESSARILY REFLECT THE OFFICIAL VIEWS OF THE SPONSOR □

LINGUISTICS RESEARCH CENTER

THE UNIVERSITY OF TEXAS

BOX 7247 • UNIVERSITY STATION • AUSTIN 12, TEXAS

AL 000577

U.S. DEPARTMENT OF HEALTH, EDUCATION & WELFARE
OFFICE OF EDUCATION

THIS DOCUMENT HAS BEEN REPRODUCED EXACTLY AS RECEIVED FROM THE PERSON OR ORGANIZATION ORIGINATING IT. POINTS OF VIEW OR OPINIONS STATED DO NOT NECESSARILY REPRESENT OFFICIAL OFFICE OF EDUCATION POSITION OR POLICY.

SYSTEM DESIGN FOR

COMPUTATIONAL LINGUISTICS

Ronald W. Jonas

"PERMISSION TO REPRODUCE THIS [redacted] MATERIAL HAS BEEN GRANTED BY RONALD W. JONAS

TO ERIC AND ORGANIZATIONS OPERATING UNDER AGREEMENTS WITH THE U.S. OFFICE OF EDUCATION. FURTHER REPRODUCTION OUTSIDE THE ERIC SYSTEM REQUIRES PERMISSION OF THE [redacted] OWNER."

LINGUISTICS RESEARCH CENTER
The University of Texas at Austin
Box 7247, University Station
Austin, Texas 78712

WA-1 LRC 67

June 1967

ABSTRACT

This paper is the first in a series presenting the features of systems designed for computational linguistics at the Linguistics Research Center. Ongoing research is expanding the application of these systems to include not only natural language but programming languages as well. From the full range of systems, this paper reports and discusses the features of current parsers, synthesizers, and translators. The fullest treatment is given to the following: (a) types of grammar most suitable for language description, (b) grammatical description of discontinuities, and (c) interlingual mapping. Alternatives are suggested and choices are made as to which features are most likely to benefit research in computational linguistics.

SYSTEM DESIGN FOR COMPUTATIONAL LINGUISTICS

Members of the Linguistics Research Center staff have been developing computer systems for the manipulation of language since 1959. The Center, organized with the primary goal of achieving machine translation of languages, has broadened its research scope to embrace the more general field of computational linguistics.

The types of systems developed at LRC include:

1. Parsers -- Programs which yield the structure underlying the strings of a language, given the grammar for that language.

2. Synthesizers -- Programs which produce the strings of a language, given its abstract structures and grammar.

3. Translators -- Given the topological properties of language in the form of specialized grammar, these programs perform the interlingual mapping necessary to translate languages. Because mapping is from abstract structure to abstract structure, the complete translation process also requires the services of parsers and synthesizers.

4. Automatic Classifiers -- Programs which will find grammatical classes for given raw text have

been under development. These are intended to code automatically the grammars for various languages.

5. Information Storage and Retrieval -- Generalized systems have been developed to manipulate data, given the associated classificatory symbols.

6. Automatic Abstractors -- Programs are planned which will perform homomorphic transformations on the output structure of present parsers. Depending upon the degree of reduction, the result will be abstracted text or classificatory symbols suitable for data storage and retrieval [7].

7. Concordance -- Systems have been developed to display text and grammars as aids in linguistic study. More generalized report generators are planned for future use.

These systems have been used solely for the manipulation of natural language. Research into the use of these systems for the processing of artificial language, particularly programming languages, is now underway. Preliminary results suggest that both computer logic and programming languages can be successfully described and manipulated with the available systems.

LRC systems are now being re-programmed to operate on the new Control Data 6600 computer installed at The

University of Texas at Austin computation center. This re-programming task has provided an ideal opportunity for the LRC systems group to improve certain features in the programs for language manipulation. We have chosen a set of features which appear to be the most desirable for CD 6600 programs. This paper will specify these features for the parsers, synthesizers, and translators and also give theoretical and experimental justification for the choice. Features chosen for the remaining systems will be discussed in subsequent papers. We assume here a general understanding of the LRC model as it has been designed in the past and discuss only those features we propose to change [10, 11, 12, 13].

The LRC model will be explained in terms of the transformational theory of grammar, for there appears to be a more widespread understanding of this theory than of the model explained in its own terms. We do not imply, however, that the LRC system is an implementation of transformational theory. The LRC model differs in known ways from a transformational model and those differences will be noted. It would, on the other hand, be relatively simple to provide the LRC model with certain alternative features, causing it to behave like a transformational system [3, 5]. We plan to augment the model in this way in the future.

The transformational model has two major components: base and transformations. The base specifies "deep", perhaps universal, properties of language, while transformations are concerned with "surface" properties (i.e., the mapping of deep structures into utterable strings of some particular

language). If we think of the transformational model as primarily generative, the base may be regarded as the first phase of the process, generating structural trees explicitly stating basic (deep) properties of language. These trees have terminal nodes, but they do not necessarily form a string which is meaningful in any particular language. The representation used so far in the base has been an ordered context-sensitive grammar. Context-sensitivity and ordering are quite independent of each other, but together they control the type of deep structures output from the base.

The transformational component, the last phase of the generative process, is expressed as a grammar having ordered rules in a highly specialized format. These rules specify how to take particular basic trees and reshape them so the terminals of the resultant trees are in the desired order. The resultant terminals must form a string, reading left to right, which belongs to the language for which the transformations are written. This basic tree is one example:

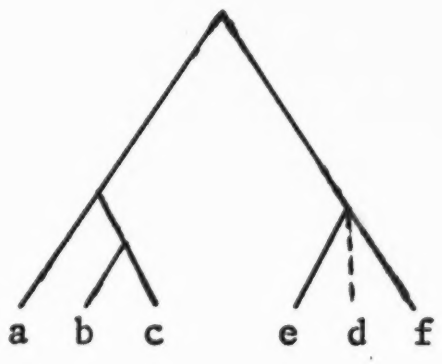


Figure 1

If string abcedf is not a string of language X, then a transformation may be specified to yield some surface tree with string abcdef belonging to the language X:

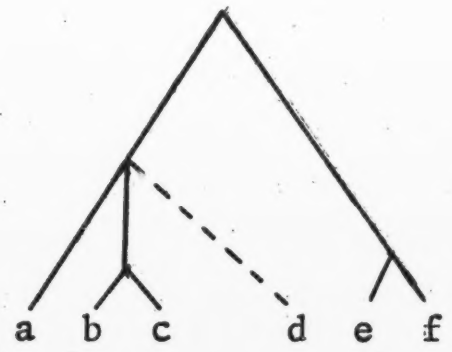


Figure 2

The trees generated by the base are regarded by Bach [2], among others, as being language-universal statements. Transformations are regarded as a mapping from the universal statements into language-specific statements. It has been shown for certain languages that dialectal differences can be entirely accounted for at the transformational level. That is, given the same deep structures and the same transformational rules, dialects may be defined simply by reordering the application of these rules. This is highly speculative, of course, but it does provide a framework in which to discuss the LRC model.

The generative process of transformational theory is roughly analogous to synthesis in the LRC model, although the two processes are dissimilar in some ways. Both the base trees and the surface trees are in an explicit form in the LRC model. In the generative model, however, the

base trees are explicit but the surface trees are not. The basic trees, together with the appropriate transformations, form only an abstract surface tree; that is, the final output of the generative model is not a tree like Figure 2. Rather it is a tree like Figure 1, plus a notation about where to move the branch leading to d. The mapping of basic trees onto surface trees is as minimal as the specified transformations. No explicit mapping is specified for those parts of the trees which are common. For an explanation of the difference in these two processes, it is necessary to look at parsing, the reverse process, in the LRC model.

Given an input string, the parser must first find some surface tree which fits the string. Certain input strings may be ambiguous, causing this process to find more than one surface tree to fit the input. These alternate surface trees are presumably all the ones which some generative process could have produced to yield the given string. Thus, the first step in analysis is to write a grammar which can be used by the process to discover all (and only) the surface trees for each input string.

The next step of analysis maps surface trees (found in the first step) into the basic trees from which they might have been generated. In terms of the examples of Figures 1 and 2, this means:

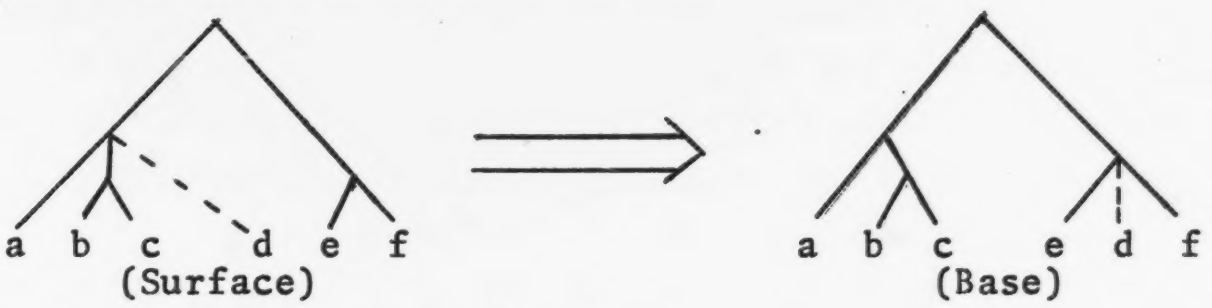


Figure 3

The resultant base tree is the desired analysis output. It is the structure which presumably has universal, or interlingual, properties that lend themselves to interlanguage translation.

Interlingual mapping has occurred in the LRC model in two mutually exclusive modes: syntactic or semantic. The interlingual mapping of base trees onto base trees has been the semantic mode; the interlingual mapping of surface trees onto surface trees has been the syntactic mode. Conceptually, the two modes operate identically, differing not so much in operation as in the level of tree mapped.

The parallelism of the two modes of mapping has existed because the form of the base and surface trees has been the same -- both explicit. Syntactic translation has been achieved by mapping the explicit surface trees found in the first step of analysis directly onto equivalent explicit surface trees of the output language. Semantic translation has involved the mapping of explicit surface trees of the input language, followed by the direct mapping of the explicit base trees of the input language onto equivalent explicit base trees of the output language. It is the resultant base trees which the synthesis process uses to produce output.

Potentially, this method causes a large duplication of descriptive and operational effort in interlingual mapping. Syntactic translation is minimal enough, but the semantic variety is quite another matter. It is probable that the base and surface trees associated with a particular string of some language have a number of common substructures. That

is, transformations generally do not alter every part of a surface tree in the process of converting it to a base tree (and vice versa).

Because the LRC model has been designed in the semantic mode to map only explicit base trees interlingually, it has been necessary to reconstitute all the parts of the input surface trees into the input base trees before interlingual mapping. For semantic translation this means duplicate mapping from surface to base trees and from base trees to interlingual classes for all structure which is common to base and surface trees. This pair of mappings duplicates the interlingual mapping specified for syntactic level translation, creating much unnecessary work. A straightforward simplification can be made.

The transformations of the Chomsky model detail a mapping from base trees onto surface trees for only those parts of the two trees which are not identical, creating a rather economical statement of structure. As mentioned previously, surface trees are abstract. A minimal statement is achieved for the generative process with the explicit base trees and a set of transformations which would yield a surface tree. The mapping of base structure into identical surface structure is not explicitly stated. This kind of minimal statement has value for the LRC model.

It appears that surface trees must be in an explicit form in the analysis process[15]. Given this requirement, economy may be achieved by specifying a set of transformation-like operations to define abstract base trees.

This specification will affect the way in which semantic quality translation operates, but not syntactic. In syntactic translation, the first phase of input language analysis will continue to use a surface grammar to find explicit surface trees. These trees will still be interlingually mapped onto equivalent output language surface trees, followed by synthesis of output strings.

Second order parsing and interlingual mapping will be changed for semantic translation. The LRC model has required a grammatical interface between the first and second orders of description. That is, all parts of the surface structure have had to be redefined as second order primitive classes. This requirement has been a prerequisite for mapping surface trees into explicit base trees. An undesirable side effect has been the necessity to re-state all surface structure in an interface grammar (semantic level dictionary rules), resulting in a rather large number of one-member second order primitive classes. Such classes have merely re-named the already-resolved surface structure just to meet the interface and explicit base tree requirements of semantic translation.

Having decided to use abstract base trees, we can relax the definition of the second order. The second order grammar may have complete and direct access to the surface structures found by the first order. Thus, first order structure may either be directly referenced by second order rules or it may be classified into more inclusive second order terminal rules before higher level second order rules are applied. These terminal rules would be

functional intermediate classifications defining whole classes of surface structure upon which a higher second order rule may operate, in contrast with the wholesale reclassification previously required in the inter-order interface.

Permitting the second order to reference the first order directly suggests a minimization in the grammar. That is, the second order grammar need only operate upon those parts of the explicit surface structure which must be changed. Correspondingly, interlingual mapping would be provided only for that base structure mapped from unlike surface structure, but not for structure which remains the same. The latter could be mapped with the same interlingual mappings used for syntactic translation. There would be no reason to duplicate the interlingual mappings for unchanging structure. Therefore, semantic quality translation would involve transfer of both second-order structure and remaining (unchanged) first-order structure.

There is no particular reason to posit two orders of interlingual mapping. Instead we may think of translation as a collection of mappings from either or both orders of structure. If we decided not to run the second-order parser, the input string could only be resolved into a surface structure. Correspondingly, only those mappings for surface structures could be involved in the interlingual mapping process, resulting in syntactic translation. If we chose to run the process through both orders of parsing, interlingual mapping would occur from both levels of structure, yielding the equivalent to what we have been calling semantic

translation. Both descriptively and operationally it would be more efficient.

Additional features might be needed to accomplish "semantic" translation. The rules used in the LRC model for the surface grammar have been context-free and unordered. They have not been exactly context-free, however, for certain operators available on the terms of the rules have made this a more powerful kind of grammar. These surface grammars assign superscripts to the surface structure so the input elements may be reordered. A second-order grammar, also roughly context-free and unordered, presumably resolves a little more of the structure. There are many kinds of grammars with which to describe languages and there is a great deal of debate among linguists as to which ones are the most suitable for describing natural languages. Some formal statements are available concerning grammars which are unnecessarily restrictive and therefore unsuitable for describing natural language. There is no statement, however, about which of the remaining grammars is best. Most of the known grammars have been put into a hierarchical arrangement which tells something about their relative powers for description, as follows:

Unrestricted Rewriting System

.

.

.

Context Sensitive Grammar

Context Free Grammar

.

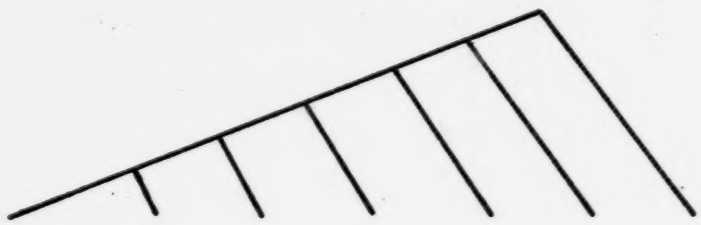
.

.

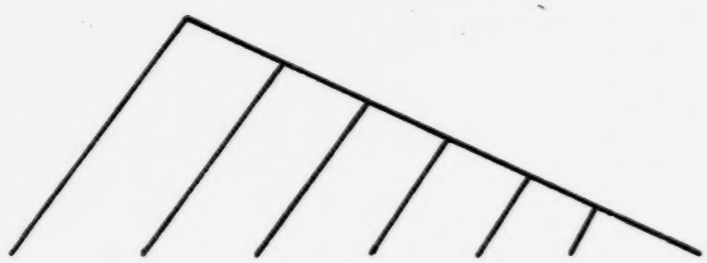
.
.
.

Meta-linear Grammar
Linear Grammar
One-sided Linear Grammar

The most powerful type of grammar is an unrestricted rewriting system, which can rewrite any set of symbols with any other set of symbols. At the other end of the scale is something called a one-sided linear grammar, which is perhaps the weakest. A left one-sided linear grammar generates structures of the following type:



A right one-sided linear grammar generates structures such as:



Neither of these structures, in itself, is adequate for describing natural languages.

The power implied by the hierarchical arrangement above is only one of two ways in which grammatical power can be measured. The grammars near the top of the scale have a greater weak generative capacity. That is, they are able to account for more strings of more kinds of languages than those at the bottom. Strong generative capacity increases as it approaches the bottom of this hierarchy. Grammars with strong generative capacity assign the fewest alternate structures to the strings of a language. Correspondingly, the grammars at the bottom of this hierarchy are so restrictive that they assign only one structure per input string.

Somewhere between the extremes is the grammar best suited to natural language description. LRC has used context free grammars, whereas transformationalists have used context sensitive grammars for the base. However, the grammars used at the MITRE Corporation for transformational surface parsing have been context free [15]. MITRE linguists observe that CF grammars generate quite a few more alternate structures than they prefer. For this reason they have been experimenting with CS grammars in order to reduce the number of alternate surface trees generated by the input grammar.

According to the measure of strong generative capacity, CS grammars should generate more alternate surface structures than CF grammars. Why, then, should MITRE be considering the use of CS grammars for controlling alternate structures? The answer to this apparent paradox lies in the fact that formal statements about generative capacity

can only be made in general. It is, of course, possible to write a CS grammar which is simply a re-statement of all the features represented in a CF grammar. In such a case, it would be impossible to claim that either of these equivalent grammars had more or less weak or strong generative capacity. Such CS grammars, which are exceptions to the general statement, will be instrumental in increasing the strong generative capacity in the description of particular languages.

In general, CS grammars may impose controls on the rewriting of strings (by the use of context) which permit more languages to be described than CF grammars. The cost of such power is an increase in the number of alternate structures, depending upon the kinds of things to be accomplished with context sensitive grammars. That is, we may write a rule which has more restrictive context for one purpose and less restrictive context for another purpose. The two together may provide enough additional power to describe more strings than a context free grammar. However, if the more restrictive context of one such rule is included within the less restrictive context of another rule, overlapping structures may occur. It is possible to reduce this overlap or to eliminate it by assuring that the grammar has the same restrictiveness or lack of it in every rule, i.e. the same number of context terms. The result of such a restriction is, in effect, a context free grammar. By increasing its strong generative capacity we have decreased its weak generative capacity and reduced the number of strings for which the grammar is able to account.

The alternation described above is peculiar to context sensitive rules. There is a second type of alternation,

common to both CF and CS grammars. Two completely independent sections of the same grammar may account for the same strings in different ways. To the extent that rules from one of these sections can be discarded, there is outright duplication in the grammar -- an accounting for something already accounted for. This alternation (duplication) is controllable because the grammar may be improved by discarding the offending section of rules. The first kind of alternation is not necessarily controllable; it occurs when there are two irreducible sections of grammar only a common subset of whose strings are duplicately accounted for. Discarding one section of rules would eliminate alternation, but it would also eliminate the structure for the non-alternating strings. The only solution is to rewrite both sections of the grammar to eliminate the overlap. If CS rules are required to account for the strings, it may not be possible to eliminate the overlap completely.

Now let us consider the matter of mixed grammars. Start with a completely context sensitive grammar of some language; then divide the grammar into two parts, A and B. Into part A put the set of CS rules which are strongly equivalent to some set of context free rules. Into part B put the remaining rules, which can in no way be reduced to CF rules. Next, define a subgrammar A' which contains the set of CF rules equivalent to the CS rules in part A.

It would be tempting to say that part A has greater weak and lesser strong generative capacities than part A'. But this simply is not so. They are equivalent.

By the same token, a grammar comprised of A and B has identical capacities with a grammar comprised of A' and B. If we were able to write a CF grammar strongly equivalent to CS grammar A-B, we would still be forced to admit that their generative capacities are identical. Particular CS grammars may be written which are strongly equivalent to particular CF grammars.

Clearly, this proves that there may be some particular pair of grammars for which the general statement about CS generative capacities relative to those of CF does not hold. More importantly, for particular (sub)strings, we may write a CS grammar which has greater strong generative capacity than a corresponding CF grammar. For example, consider a language with a single string aaan to which we want to assign only the following structure:

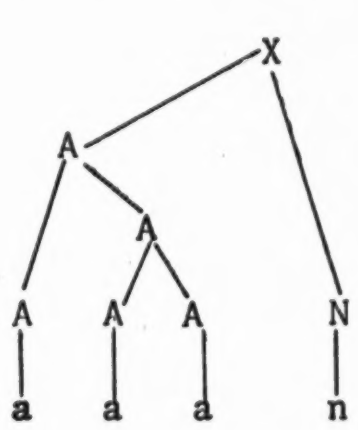


Figure 4

The following unordered CS grammar will yield only this structure:

$$a \rightarrow A \qquad\qquad AA \rightarrow A/__N$$
$$n \rightarrow N \qquad\qquad AN \rightarrow X$$

There is no unordered CF grammar which can assign only this structure. We must settle for a CF grammar like the following:

$$a \rightarrow A \qquad\qquad AA \rightarrow A$$
$$n \rightarrow N \qquad\qquad AN \rightarrow X$$

It is undeniably true that particular (sub)strings may be describable by particular CS grammars with greater strong generative capacity than CF grammars. It happens to be true in this example because rule AA → A is applicable in fewer contexts with the CS grammar than with the CF grammar. Such examples may well occur in natural language. With proper care, we should be able to partition a language into sets of substrings so that the subgrammars for these sets are mutually exclusive. It is entirely plausible that for any given set of substrings we may write a CS grammar with greater strong generative capacity than any CF grammar. If each <u>particular</u> subgrammar is written to obtain the greatest strong generative capacity, the total grammar is assured of having the greatest strong generative capacity. Any <u>particular</u> such grammar may well have a mixture of CS and CF rules.

The LRC model has the capacity to handle CS as well as CF rules. However, in the interest of establishing minimal operating systems on the CD 6600 as soon as possible, implementation of context sensitivity will be delayed. As

soon as a context free parser is available for linguistic use, an augmented version (with CS features) will be added.

One other matter to be considered is ordering versus non-ordering of the grammatical rules. Ordering places restrictions on the application of the grammar. Out of all the possible structures one could get by freely applying the rules in any order, ordering causes only a subset to be realized. Any grammar may be applied in an ordered or unordered manner. Ordering has to do only with the application process, not the type of grammar. The LRC parser uses unordered grammars, while transformationalists use ordered grammars. McCawley[8] has questioned the use of ordering in the base component of transformational grammars. At this time, the arguments for and against ordering seem equally valid. In the absence of a definitive statement, the LRC parser will continue to apply grammars in an unordered fashion.

The next problem of some importance in the model is accounting for discontinuity of elements. Its solution involves bringing together (potentially) isolated parts of the structure so they may function as a unit. There are many ways in which this may be accomplished. Transformationalists specify transformations to reshape the structure. The solution in the LRC model has been: the right-hand terms of the surface grammar rules are numbered with superscripts (using S operators), which may assign the terms a standard or non-standard order, reading left to right. When surface parsing has been completed for a given input string, derived

superscripts are calculated. This process assigns each branch of the surface tree a unique number by comparing superscripts of the branches with respect to each other. The derived superscripts define a new, standardized order for the elements of the input string. So, by the proper assignment of superscripts in the surface grammar, we are able to logically, if not physically, bring together discontinuous elements of the input string. The standardized string is then made available to second-order parsing, which is able to recognize the previously discontinuous elements with immediate-constituent rules.

There is, however, a serious deficiency in this method of handling discontinuity. Transformational theory provides the power to operate upon a tree iteratively. A series of transformations may be specified which reshape a tree many times before it arrives at its final form. Obviously with only two levels of analysis, the LRC parser has permitted only one set of transformations, not an iteration. To achieve iterative operations upon discontinuities one would need n orders of analysis, with n superscripted grammars to match. Derived superscripts would be calculated after each order, and the input string logically reordered each time.

Certainly an analysis of n orders is unthinkable. Whether we need iterative access to discontinuities at all is questionable. For the majority of cases we can probably dispense with iterative methods and simulate iterate treatment of a particular discontinuity in a single order of

parsing, which is all the LRC model has been prepared to handle. Even if we could "artifically" handle all discontinuity in this manner, superscripting would still be cumbersome. The more complex a discontinuity (the more elements there are involved in a total structure) the more difficult the task of assigning superscripts becomes. In all intractable cases, we would probably have to write separate subgrammars in order to acquire the proper derived superscripts. Such a practice would undoubtedly lead to outright duplication in the surface grammar, all for the sake of producing different orderings of the input string for second-order analysis. It is questionable whether language can be adequately described in such a manner and whether the surface parser can handle the burden of the redundant grammars.

There is no reason to handle discontinuity in this manner, for iterative or non-iterative treatment of discontinuity can be handled by application of discontinuous rules. Such rules are more disorderly than immediate-constituent rules, but they permit us to free the parser from the constraints imposed by superscripting. The application of discontinuous rules would be the primary function of the new second order. The question is: why separate the treatment of discontinuity into a special order, particularly into the second order rather than the first? We might say that "bringing together isolated parts of the structure so they may function as a unit" is in the nature of mapping surface structure into the base structure. In other words, discontinuous elements are separated because

the surface structure dictates that they should be. When we say that discontinuous elements are functioning as a unit, we are simply noting that in the basic structure they are immediate neighbors and structurally related, despite their surface relationship. Thus, second order application of discontinuous rules becomes the mapping of elements which were discontinuous in the surface structure into adjacent positions in the base structure.

How can second order use of discontinuous rules be made to function properly in the translation process? They would appear to classify isolated elements of the input without any contextual reference. How, without this reference, is the synthesis process able to decide where to put the corresponding discontinuous elements? The answer is simply that there always <u>is</u> contextual reference. Because these rules operate at the second order upon the first order structure, the surface structure gives the context. When the parser applies a discontinuous rule, it "remembers" where it found the elements and what their relationship was to all the other elements in the surface structure. The translation process preserves this information gleaned by analysis, carries it through interlingual mapping, and makes it available when synthesis is trying to construct an output surface tree.

If we now regard interlingual mapping as occurring at all levels of structure simultaneously, the interlingual transfer of discontinuity information is descriptively minimal. Consider the following equivalent examples:

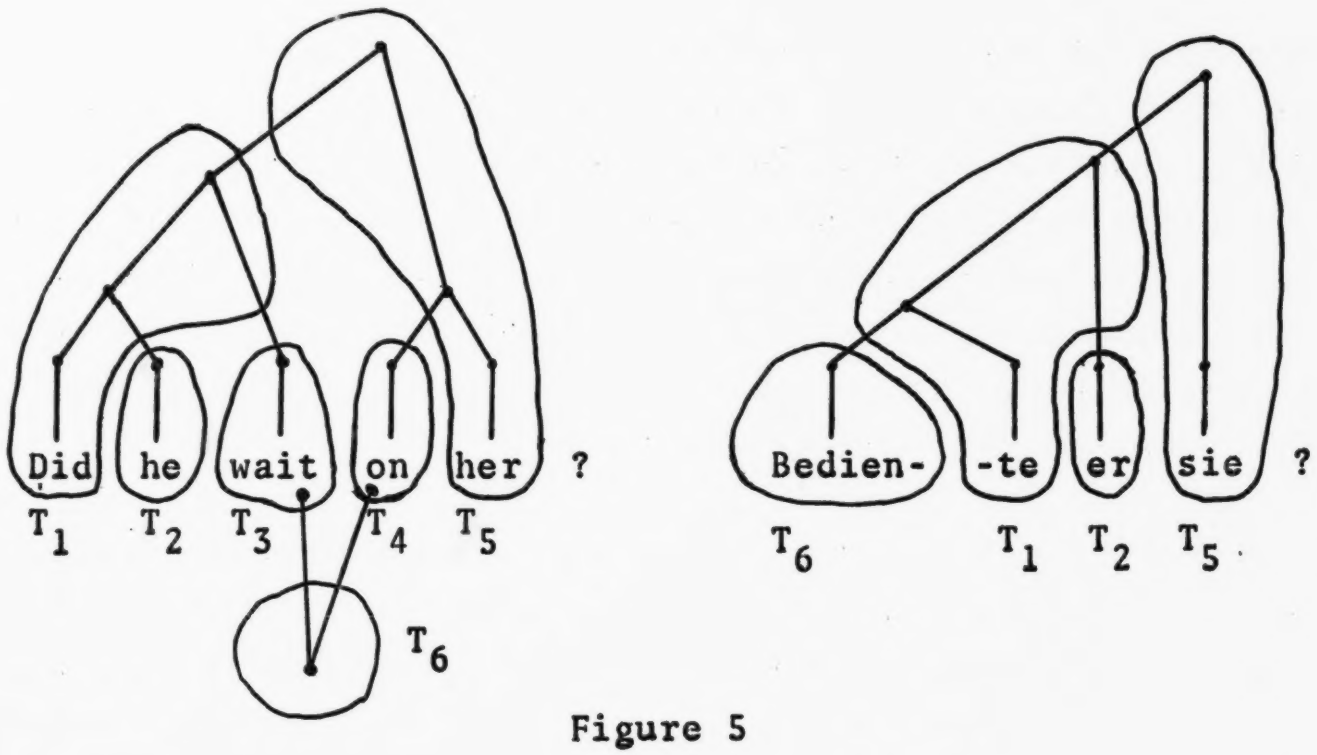


Figure 5

The structure below the left-hand example is intended to reflect second order structure. There is no reason to require that the second order resolution of wait on involve second order resolution of bedienen. The treatment of discontinuity entails minimal description so long as the interlingual mappings (T_x) are common to all structural levels and the equivalent structures are commonly named (T_6 in particular).

Discontinuous rules do not eliminate the need for superscripting in the model. Superscripts are also used to preserve information concerning connections between interlingual classes. The argument that discontinuity is better handled by discontinuous rules than by superscripts has already been advanced. A preferable scheme for maintaining interlingual class connection information will be offered

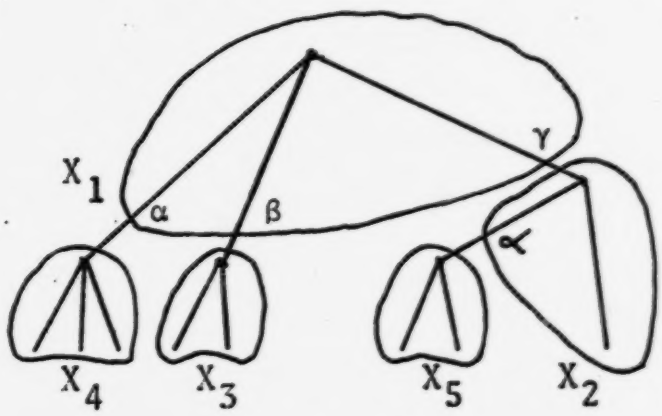
here. Thus, the use of superscripts in monolingual grammars will no longer exist as a feature of the model. Their function will be replaced by discontinuous rules and the following scheme for coding interlingual connection information in the interlingual grammars.

In the examples of Figure 5, each T_x is connected to other T_x's at particular branch points of the monolingual structure. As mentioned earlier, the set of derived superscripts computed for any such structure assigns unique numbers to each branch of the tree. These unique numbers are used in transferring connection information from one language to another. There is no reason to give a name to every branch of the monolingual structure; names are needed only at the connection points. Monolingual grammars cannot contain information about which branches meet these qualifications, but interlingual grammars can and do. Interlingual rules have always contained operators defining connection points within each class. For example, the rules for interlingual class T_1 in the examples of Figure 5 would have only two such operators. These operators tell the translation process which branch-names of the monolingual structure to "remember" as interlingual connection points. The process correspondingly remembers the (unique) derived superscript numbers of the appropriate branches of the input structure. These names become the interlingual connection points of the output structures having corresponding derived superscripts.

There is no reason for the process to "remember" a derived superscript number just to name each connection

point indicated by an interlingual rule. A unique name could be coded with each connection operator in the interlingual rule. It is not at all necessary for every connection operator in the grammar to have a unique name; connection names need only be unique within each interlingual class. In any event, putting the connection names into the interlingual rules reduces the amount of branch naming previously induced by the superscripts in the monolingual grammars. Consider the following examples:

Input Language



Output Language

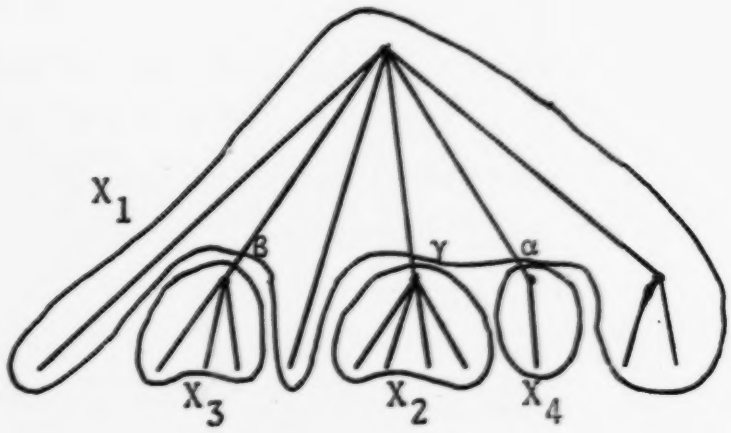


Figure 6

X's serve as interlingual class names and Greek letters as connection branch names. Interlingual parsing would assign these names from its grammar, as above, and then create connection information. This information would specify which interlingual class was connected at each connecting branch:

$$[X_1, \alpha] : X_4$$
$$[X_1, \beta] : X_3$$
$$[X_1, \gamma] : X_2$$
$$[X_2, \alpha] : X_5$$

There is no connecting information for classes X_3, X_4, and X_5 because they are terminal.

To reorder interlingual classes for the output language, it is necessary to control the assignment of connecting names in interlingual rules. Connecting names need be unique only within an interlingual class, and they are common to all languages having that class. In the example, the connection names have been assigned in such a way as to cause the dependent classes X_4, X_3, X_2 of the input language to be reordered as X_3, X_2, X_4 in the output language. This is accomplished in synthesis as follows. Starting with X_1, in the output language, synthesis may decide to connect the proper class at branch β. Looking in the table of connection information, it finds that X_3 is to be connected there. Synthesis then finds all output language structures interlingually classified as X_3 and tries to connect them at $X_1 - \beta$. Next it may decide to extend $X_1 - \gamma$, etc., until all connections have been made in the output language.

The consequences of this sequence for the descriptive process are obvious. First, we may define a set of interlingual classes which are mutually satisfactory for all languages. Then each language may be described with structures completely satisfactory to that particular

language, just so long as it is possible to map all the structural pieces of that language into the chosen interlingual classes. When the various languages have been individually described, it is merely necessary to return to the collection of interlingual classes and agree upon mutual names for the branches involved in connecting the interlingual classes. For example, in order to assure that the branches for one language structure mapped into interlingual class X_1 are in a different order with respect to those for another language, it is necessary to define a set of names for interlingual class X_1 and assign them to the connecting branches for interlingual class X_1 in such a way that the corresponding branches in the two languages have the same name.

There is one further matter to be considered with regard to interlingual mapping. In the LRC model it has been necessary for all structures mapped into the same interlingual class to have the same degree, i.e. the same number of connecting branches. If, for example, five interlingual classes are connected on the input, five have been required in the output. Thus, the problem of specifying what the process is to do when there are different amounts of structure interlingually mapped is avoided. With the new type of mapping, we can re-examine this restriction and loosen it up a bit. Consider, for example, X_5 to be some kind of distinction in the input language that is not needed in the output language. Why not write X_2 in such a way that the input language has a connecting branch named α (for X_5) while the output has no corresponding branch for

X_2? The synthesis process would then ignore X_5 because there would be no place to correct it. As illustrated, the output language would involve only four of the interlingual classes discovered for the input language.

The loss of X_5 in translation is not at all unnatural. Furthermore, as suggested in the examples, both structures classified as X_2 are well-formed. It happens that for X_2 the input language has a rule with one terminal and one non-terminal symbol, while the output language has a rule with only terminal symbols (four). A problem would arise, of course, if we wanted to translate in the reverse direction. In effect it would be necessary to acquire the distinction X_5 not made in the input language.

There are two ways to handle this problem. First, we could simply say that we do not have any such thing as a reversable translation. Rather, we write our grammars in such a way that the input language always has greater than or equal to the number of interlingual classes of the output language. Secondly, we could solve the problem in this way. If, for example, the class X_5 appears in the output, not having appeared in the input, we might let the monolingual grammar of the output language be of assistance. Synthesis is able to detect that class X_2 has a connecting branch in the output language which was not in the input. This is easily recognized because the output monolingual rule for X_2 has a connecting branch but there is no combination (X_2,α) in the connection table. This could signal that synthesis is responsible for generating certain new structure

in the output which was not in the input. In particular, the distinction made by interlingual class X_5 could be generated. The unsatisfied connection point will have a non-terminal symbol. The synthesis process can be made to generate all possible subtrees starting with that non-terminal symbol; it should generate all possible distinctions X_5. Further thought must be given to how we would want this generation to be controlled. Meanwhile, we can require that the input language always have the same or a greater number of connecting branches than the output for each interlingual class.

Finally, there is the minor matter of deciding just what is an equal number of branch names in the interlingual class being translated. To do this, we count the number of <u>different</u> connecting branch names occurring within each interlingual class for the input language and the output language and compare them. Languages with and without reflexivity may be mapped into each other more freely by reusing connecting branch names for related parts. For example:

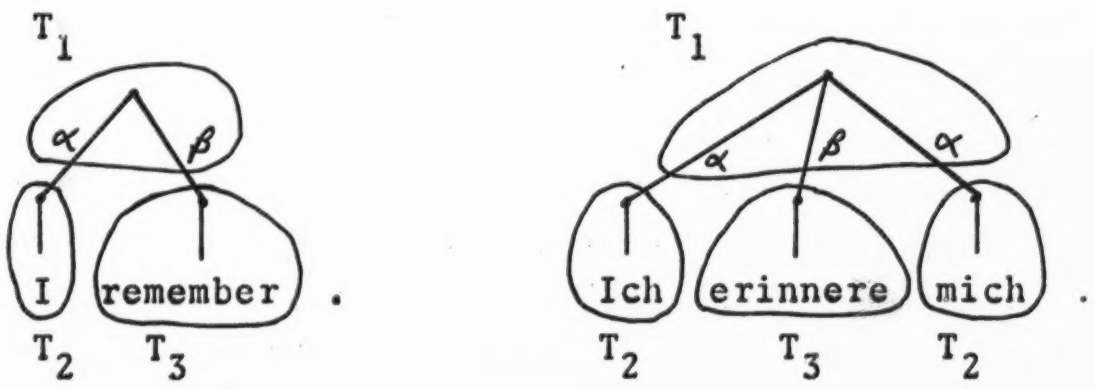


Figure 7

In this case, T_1 is considered to have the same degree (2) in both languages.

In summary, feature changes proposed for the parsers, synthesizers, and translators of the LRC model are:

1. Eliminate superscripting in monolingual grammars.

2. Cause interlingual mapping across all levels of structure, which would reduce second order structure to a minimum.

3. Code interlingual connection information on interlingual rules.

4. Introduce discontinuous rules to the second order parser.

5. Add context-sensitivity to the parsing and synthesizing grammars.

Details of the remaining processes of the LRC model will be presented in later papers.

BIBLIOGRAPHY

1. Emmon W. Bach, An Introduction to Transformational Grammars, New York: Holt, Rinehart and Winston, 1964.

2. Emmon W. Bach, "Nouns and Noun Phrases," paper presented at the Symposium on Universals in Linguistic Theory, The University of Texas at Austin, April 14, 1967.

3. Noam Chomsky, Syntactic Structures, The Hague: Mouton and Company, 1957.

4. Noam Chomsky, "Formal Properties of Grammars," Handbook of Mathematical Psychology, Vol. II, edited by Luce, Galanter and Bush, New York: Wiley and Sons, 1963.

5. Noam Chomsky, Aspects of the Theory of Syntax, Cambridge: The M.I.T. Press, 1965.

6. Jerrold J. Katz and Jerry A. Fodor, "The Structure of a Semantic Theory," The Structure of Language, edited by Fodor and Katz, Englewood Cliffs: Prentice-Hall, 1965, pp. 479-518.

7. Susumu Kuno, "A System for Transformational Analysis," MATHEMATICAL LINGUISTICS AND AUTOMATIC TRANSLATION, Cambridge: Harvard University Computation Laboratory, August, 1965

8. James D. McCawley, "Concerning the Base Component of a Transformational Grammar," Ditto (Revised), Chicago: University of Chicago, August, 1966.

9. Readings in Automatic Language Processing, David G. Hays, editor, New York: American Elsevier Publishing Company, 1966.

10. Arnold C. Satterthwait, "Programming Languages for Computational Linguistics," Advances in Computers, 7:221-225, edited by Franz L. Alt and Morris Rubinoff, New York: Academic Press, 1966.

11. "Symposium on the Current Status of Research," LRC 63 SR-1, Austin: Linguistics Research Center, October, 1963.

12. "Thirteenth Quarterly Progress Report (1 May 1962-31 July 1962)," Austin: Linguistics Research Center, July, 1962.

13. Wayne Tosh, Syntactic Translation, The Hague: Mouton and Company, 1965.

14. Victor H. Yngve, "A Framework for Syntactic Translation," Mechanical Translation, 4:59-65 (December, 1957).

15. Arnold M. Zwicky, et al, "The MITRE Syntactic Analysis Procedure for Transformational Grammars," AFIPS Conference Proceedings, 1965, (Fall Joint Computer Conference), 27:317-326.